Journal of
Sensor and Actuator Networks

MDPI

Article

A Privacy-by-Design Contextual Suggestion System for Tourism †

Pavlos S. Efraimidis [1,*], George Drosatos [2], Avi Arampatzis [1], Giorgos Stamatelatos [1] and Ioannis N. Athanasiadis [1,3]

[1] Electrical & Computer Engineering Department, Democritus University of Thrace, Xanthi 67100, Greece; avi@ee.duth.gr (A.A.); gstamat@ee.duth.gr (G.S.); iathan@ee.duth.gr (I.N.A.)
[2] School of Medicine, Democritus University of Thrace, Dragana, Alexandroupoli 68100, Greece; gdrosato@ee.duth.gr
[3] Information Technology Group, Wageningen University, Wageningen 6706 KN, The Netherlands
* Correspondence: pefraimi@ee.duth.gr; Tel.: +30-254-107-9756

† A preliminary version of this work appeared as "Pythia: A Privacy-Enhanced Personalized Contextual Suggestion System for Tourism. In the Proceedings of the 39th IEEE Annual International Computers, Software & Applications Conference (COMPSAC 2015), Taichung, Taiwan, 1–5 July 2015.

Academic Editor: Ioannis Chatzigiannakis
Received: 21 December 2015; Accepted: 22 April 2016; Published: 5 May 2016

Abstract: We focus on personal data generated by the sensors and through the everyday usage of smart devices and take advantage of these data to build a non-invasive contextual suggestion system for tourism. The system, which we call Pythia, exploits the computational capabilities of modern smart devices to offer high quality personalized POI (point of interest) recommendations. To protect user privacy, we apply a *privacy by design* approach within all of the steps of creating Pythia. The outcome is a system that comprises important architectural and operational innovations. The system is designed to process sensitive personal data, such as location traces, browsing history and web searches (query logs), to automatically infer user preferences and build corresponding POI-based user profiles. These profiles are then used by a contextual suggestion engine to anticipate user choices and make POI recommendations for tourists. Privacy leaks are minimized by implementing an important part of the system functionality at the user side, either as a mobile app or as a client-side web application, and by taking additional precautions, like data generalization, wherever necessary. As a proof of concept, we present a prototype that implements the aforementioned mechanisms on the Android platform accompanied with certain web applications. Even though the current prototype focuses only on location data, the results from the evaluation of the contextual suggestion algorithms and the user experience feedback from volunteers who used the prototype are very positive.

Keywords: privacy; personalization; contextual suggestion; privacy by design; non-invasiveness; tourism; mobile computing; recommendation systems

1. Introduction

Smart devices, and in particular smartphones, are ubiquitous devices much more than communication tools, as they combine features of a cell phone along with computing functionalities. Ubiquitous devices mingle and sometimes interfere with a person's everyday life. Smartphones hold on to a plethora of personal, yet heterogeneous data generated from various software applications, hardware components and embedded sensors, constituting a very rich set of private information. Consequently, a smartphone can be considered as a well-informed, about its owner's interests, ubiquitous digital companion that comprises a diverse set of sensors and strong computational and networking capabilities. Thanks to these features, information available to a smartphone can be used in many circumstances to assist its owner with suggestions and recommendations.

In this work, we focus on the sets of personal data of smart devices and consider how they can be used to make touristic contextual suggestions to their owners. Whatever may attract a tourist can be considered a point of interest (POI), and tourists often are overwhelmed by the large number and variety of POIs in the places they visit. Typically, a tourist has to make his or her choices in a short time and without being fully informed about his or her options regarding the available POIs. Assuming that a contemporary tourist owns a smartphone equipped with a number of sensors and is familiar with mobile applications (apps), mobile recommender systems can become valuable tools. In fact, mobile tourism services are currently a very active research topic; for a collection of related papers, see, e.g., [1].

A recommender system for tourism aims to advise its user on which POIs to visit while staying in a certain area. Such suggestions are produced based on inputs, and a simple approach would be to ask the user about his or her preferences and to include information, such as user needs, interests and constraints, enter those into some system and then have the system correlate these preferences with POIs that have similar parameters [2]. Another option is to use additional input evaluations and ratings of other tourists with similar interests in a collaborative filtering fashion [3]. Additionally, travelers who are in close spatial and temporal proximity often share common travel interests or needs in a crowd-sourced manner [4,5]. A common requirement of all of these approaches is that users have to enter *themselves* their personal information and build their profile into some system. In this paper, we propose an approach to automatically build a POI-based user profile and show how this profile can be used as input to a contextual suggestion mechanism.

Another, possibly more important, requirement of existing recommender systems is that user profiles have to be stored and managed by the recommender service. User profiles, however, contain personal data, some of which may be considered *sensitive personal data* according to the Data Protection Directive 95/46/EC [6], the current draft of the forthcoming General Data Protection Regulation (GDPR) [7], which regulates the processing of personal data within the European Union, and similar regulations, which hold in other regions. This, in turn, raises privacy concerns for potential users of such systems. Moreover, the privacy issues can be further aggravated when user profiles are combined among recommender systems in order to expand the data pool and enable more intelligent recommendations [8]. The privacy concerns are even more serious in mobile recommender systems, due to the wide range of sensitive personal data (e.g., location) that are available to mobile platforms and can be accessed by mobile apps and transparently be uploaded to remote servers. In fact, user awareness of threats against location and identity privacy aspects has been recognized as one of the greatest barriers to the adoption of context-aware services [9].

In this work, we employ a *privacy by design* approach to address the privacy issues of the proposed recommendation system. Privacy by design generally refers to a holistic approach for handling the privacy issues within each step of the design, implementation and operation of a system (see, for example, [10] and the references therein). For the Pythia system, a very important architectural choice we made under the privacy by design approach is to manage the user data at the user side. We will show that the overall design of Pythia indeed enables strong and effective privacy guarantees for the users.

The exact challenge addressed in this work is to propose a privacy-enhanced, non-invasive contextual suggestion system for tourism. The targeted functionality of the system is in line with the Contextual Suggestion Track of the Text Retrieval Conferences (TREC) [11] and is about making suggestions to the user considering as context only the user's location, as well as user interests via personal preferences and past history. In other words, the recommendation focuses on a single, but common situation:

> A user with a mobile device with limited interaction, but some sort of a user profile, who is in a strange town and who is looking for something to do. There is no explicit query; the implicit query is: *Here I am, what should I do?* [12,13].

In the future, context may be enriched to include, for example, temporal information (time, day, season of the year, *etc.*), weather conditions or other contextual information [14,15].

An important additional requirement (beyond TREC's definition) addressed in this work is the protection of user privacy. More precisely, we require that no personal data should be leaked to any party, including the service provider. Finally, the non-invasiveness property requires the system to operate in the background, to automatically infer the user preferences and gradually build the user profile without requesting the user's attention.

The Pythia system proposed in this paper is a privacy-enhanced contextual suggestion system for tourism, which satisfies the above challenge. The presented system innovates in several fronts by design and implementation choices to guarantee non-invasiveness and privacy preservation. We consider this combination of features, which, to our knowledge, is unique in the field of mobile recommendation systems, the main novelty of our work. More specifically, Pythia offers:

- *User-centric architecture*: Pythia is based on a user-centric architecture in which all user data are stored at the user-side. The contextual suggestion algorithms are also executed at the user-side. This design offers the users a high level of control over the system and their personal data.
- *Privacy-preserving recommendations*: The system is designed so that no personal data are disclosed to any party, including the recommender service itself. Moreover, Pythia ensures strong privacy guarantees for the user, without relying on trusted third parties and data obfuscation or heavy cryptographic techniques.
- *Rich user profiles*: As the profiles and the contextual suggestion engine reside on the user's mobile device, the system has access to his or her digital trace, which comprises a virtually unbounded, in size and type, set of personal data. In the current design, Pythia generates POI-based user profiles automatically from the raw location data.
- *Non-invasive operation*: The Pythia components operate non-invasively, without requiring user interaction. The user profile representing the user's interests is automatically built and updated.

Note that while the contextual suggestion engine has access to the local user profile, which may contain a very rich set of personal data, the suggestion algorithms base their computations only on a single profile, in contrast to collaborative filtering approaches. The comprehensiveness of the user-owned profile can counterbalance the fact that only single-user profiles are used in computing recommendations.

The rest of this paper is organized as follows. In Section 2 we discuss our approach with respect to past related work. Section 3 presents the Pythia system architecture with its components, while Section 4 details the algorithmic aspects of these components. Section 5 describes the implementation of the Pythia system. The results from the evaluation of the contextual suggestion algorithms and the user experience feedback from volunteers who used the prototype are presented in Section 6. We conclude in Section 7 with a discussion and directions for future research.

2. Related Work

2.1. Mobile Recommender Systems

There is a rich literature on recommendation systems and mobile recommendation systems; see [16,17] and the references therein for general recommendation systems. The application of recommender systems in the tourism domain has also attracted much interest from the scientific community; see, for example, [18,19] for up-to-date surveys of the field and related topics. More specifically, in [18], the authors focus on systems available for tourism applications, analyze different user interfaces that identify them and discuss various functionalities offered by these systems.

An interesting application for tourists is the myVisitPlanner^{GR} system [20], which recommends activities to visitors along with appropriate schedules to carry out these activities during a visit. The protection of privacy is also discussed, and even though users have to disclose some personal

data to the myVisitPlanner^{GR} system, the amount of personal data disclosed is kept at a low level, measures are taken to avoid accidental data leaks and the privacy policy of the application is clearly documented. Another recent work is the iGuide system [21], which aims at enabling a socially-enriched mobile tourist guide service where its recommendations are enriched with multimedia content. In this direction, in [22], the authors propose a novel approach for multimedia recommendation and annotation.

However, as stated in [19], concrete mobile recommender system methodologies for protecting user anonymity and privacy are required. Those methodologies should guarantee the effectiveness and accuracy of recommendations without compromising the privacy of user profiles and sensitive contextual information, and this work contributes towards this direction.

2.2. Privacy-Enhanced Recommender Systems

One approach to enhance privacy in recommendations systems is to apply appropriate cryptographic or algorithmic techniques. For example, two illustrative works [23,24] propose the use of homomorphic encryption as a structural element for their systems. Another system that uses randomized perturbation instead of encryption has been proposed in [25]. In mobile recommender systems for tourism, the authors in [26] propose the use of differential privacy methods to extract statistics about users' preferences for POIs, where the actual recommendations are generated by querying those statistics in order to formally enforce privacy.

In our view, the user-centric architecture of Pythia combines a set of features that is unique with respect to related work; it is simple and practical since it does not rely on trusted third parties or heavy cryptographic protocols; it is accurate since it does not need to introduce noise into the data (to protect privacy); and it offers strong privacy guarantees since the personal data are not disclosed to any party, including the recommender service provider.

2.3. Non-Invasive Recommender Systems

In [17], the authors classify recommender systems, according to the way user preferences were acquired, as *explicit* and *implicit* preference learning. The concept of non-invasive profile building in this work is similar to other related studies in implicit preference learning in the sense that the system silently monitors the users' behavior, such as POIs visited or web sites read, and progressively builds their profile. While we offer ways for a user to manually input feedback into the system, our main focus is acquiring this information in an automatic way.

Similarly, in [27], research on how recommender systems learn the user preferences (feedback gathering) and how they process them is presented. Examples of techniques for learning user profiles include the naive Bayes algorithm and the Rocchio method; the latter is also the basis for our recommendations.

An early interesting example of a non-invasive application that assists a user is the Letizia agent [28] for web browsing. Letizia tracks the user's browsing behavior and then attempts to anticipate web pages of interest for the user. Another related work is [29], where a non-invasive approach to construct web user profiles is presented. This work, too, is about the browsing behavior of the user and estimating the user's interest of a web page.

The Pythia system is inspired by the above works, but in a different context and, most importantly, in the field of smart mobile devices, where the digital trace of common users is far more comprehensive and revealing about the user's interests.

3. An Overview of Pythia

Pythia is a new privacy-enhanced user-centric contextual suggestion system for tourism. The system exploits the user's digital trace to automatically generate and update a profile, which is safely kept at the user side. A contextual suggestion component, which is also executed at the user's side, processes the profile and a location of interest and generates POI suggestions.

An overview of the Pythia system architecture is shown in Figure 1. The system comprises mobile, web, server-side and cloud components. Personal cloud storage serves as intermediate storage for the application components, and a POI collection framework manages the available POI data of the system. For the user, the most essential part of the system is Pythia's client software, which is implemented both as a mobile app and as a web app. The mobile app consists of the following components:

1. *Content collector:* A non-invasive, background service that collects (sensitive) personal data of the digital trace generated mainly by the user's mobile device. There are several kinds of personal data that can be collected. In this work, we focus on location traces, browsing history and web searches. Optionally, the collected data can be uploaded in encrypted form to the personal cloud storage of the user.
2. *POI-based profiler:* A key component of Pythia, which uses the raw personal data of the user to build a POI-based profile of his or her interests. This profile is actually a list of rated POIs that represent the user's interests. The list and the ratings are automatically generated.
3. *Contextual suggestion engine:* This component uses the POI-based profile of the user to generate a personal query, which is then used to retrieve the personalized POI suggestions for the user-specified location of interest.

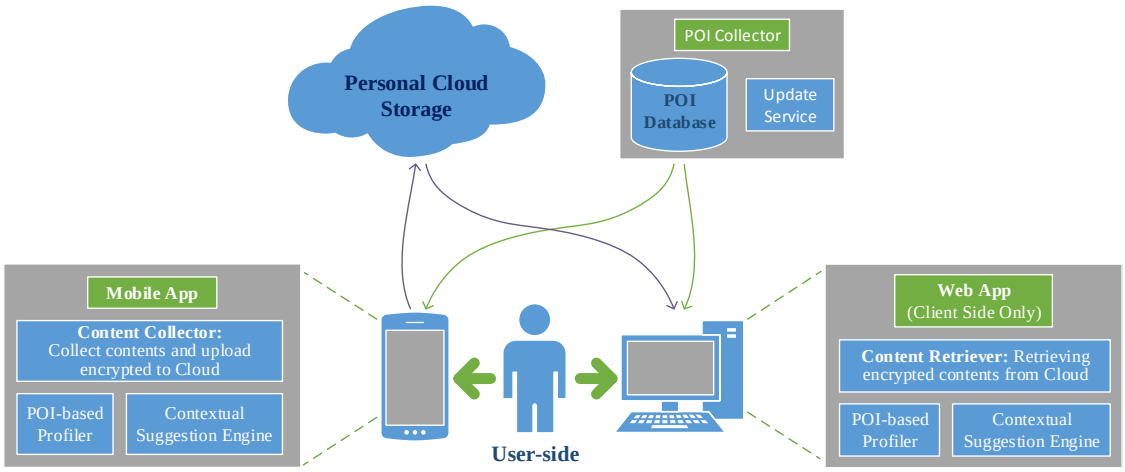

Figure 1. Overview of the architecture of the Pythia system.

The latter two components are also implemented within Pythia's web application and can be used from a browser. An additional component of Pythia's web software is:

1. *Content retriever:* A tool that retrieves the encrypted personal data from the personal cloud storage of the user. The data can then be used by the POI-based profiler of the web app.

Finally, there are two supporting components, the user's personal cloud storage and the POI collection framework.

1. *Personal cloud storage:* In principle, any popular cloud provider, such as *Google Drive*, *Microsoft OneDrive*, *Dropbox*, or some self-hosted platform, like *OwnCloud* [30] can be used. For simplicity, the current choice is *Google Drive*, since any Android device has by default access to it. This component is used for:

    - *Aggregation of raw personal data from different devices:* When a user collects content from more than one mobile device, the personal cloud storage is the common storage space where each device uploads its raw personal data. To protect user privacy even from the cloud provider,

the data uploaded to the cloud are encrypted. In particular, we use the AES [31] symmetric encryption algorithm to encrypt the data.

- *Flow of data between the application components:* The cloud storage is used by Pythia's client-side software to access the raw personal data generated on the mobile device(s).
- *Backup of the application data:* In case of data loss from some mobile device, the data stored on the personal cloud space is preserved.

2. *POI collection framework:* A central component of the Pythia system, running at the server-side. The framework comprises a POI database, a service to automatically update its contents and a web API. The POI database maintains a collection of POIs from selected areas in several countries. The update service continuously populates and updates the database with data retrieved from two popular place search engines, *Google Places* [32] and *Foursquare* [33]. The POI database contains the POIs that are used by the POI-based profiler and the contextual suggestion engine.

4. How Pythia Works

In this section, we describe how the Pythia system works, with emphasis on the algorithmic and computational aspects of its components. The content collector and the POI collection framework are responsible for collecting the data that are used by the POI profiler and the contextual suggestion engine. The content retriever and the personal cloud storage support the data flow between the application components. The way the independent components interact is shown in Figure 2.

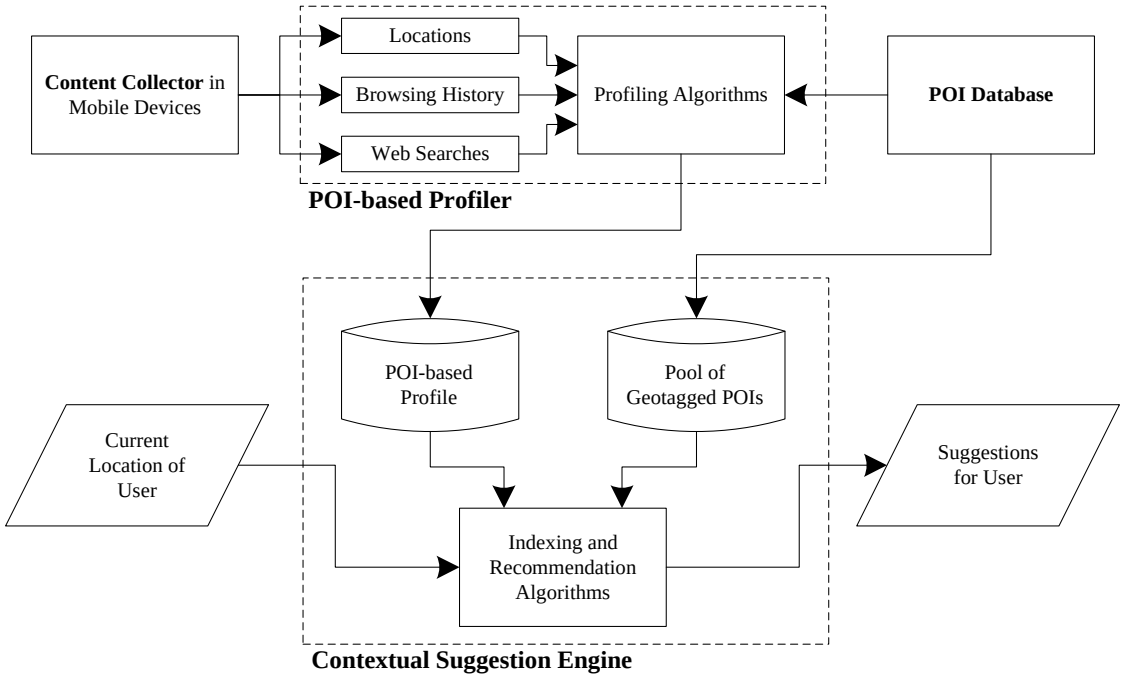

Figure 2. Interaction of the Pythia components.

4.1. Personal Data Collection

As noted earlier, the content collector collects personal data of the digital trace generated by the everyday usage of the devices. These data are stored locally on each device. Currently, we focus on the following types of personal data.

- *Location traces:* The content collector records the location of the mobile device periodically, with a period t_ℓ (e.g., $t_\ell = 5$ min). The location is requested from the GPS provider of the device. In case

the GPS provider fails to coordinate (for example, if the user is inside a building), the location is requested from the network provider (the network provider determines location based on the availability of cell tower and WiFi access points). If both location providers fail to retrieve location information, the content collector temporarily reduces the frequency of location data requests to avoid unnecessary power consumption.

- *Browsing history:* The browsing history of the user is collected with a content observer that is triggered when the user visits a web site with a mobile web browser. For each visited URL, a browsing history record is kept with the following data fields: title, url, number of visits and time of last visit.
- *Web searches:* Web searches are a special type of URL and can reveal important information about the user's interests. The content collector detects and records the web search queries submitted by the user. For each web search, the query and the submission time are recorded.

Additionally, the current version of the content collector can optionally collect data from the sensors of mobile devices, such as barometric pressure, ambient temperature, relative humidity, ambient light, acceleration, rotation and geomagnetic field. These data, however, are presently not used in Pythia.

4.2. POI Collection Framework

The POI collection framework implements a web API, which is used by other Pythia components to access the POI database. POIs can be retrieved either one by one or as chunks corresponding to a specific county or area.

The following data are available for each POI: title, geo-location (*i.e.*, geographical coordinates of location), address, phone number, categories, several URLs (e.g., website, Foursquare URL, Google+ page, *etc.*), rating, total unique (physical) visitors and total visits (as measured by Foursquare) and a collection of terms describing the POI. The POI data are obtained from two popular place search engines, namely Foursquare and Google Places, with Foursquare as the main source of information. The update service continuously updates and extends the database with new information about POIs in the defined areas of interest.

4.3. POI-Based Profiling

The raw data used by the POI-based profiler consist mainly of a series of locations the user has been, plus the browsing and web search history. These data are processed in the following steps (Figure 3):

1. First, the location traces of the user are used to extract significant places, which are defined as places where the user spends a significant amount of time and/or visits frequently. The extraction process is based on a *time-based location clustering* algorithm.
2. Then, the extracted significant places are matched (if possible) with existing POIs of the POI database. The matching is based on distance and popularity criteria of each particular significant place and the nearby POIs.
3. Next, the user's rating for each matched POI is estimated. This is also done automatically, using the user's number of visits and the average number of visits per user to this POI.

These steps are described in more detail, next.

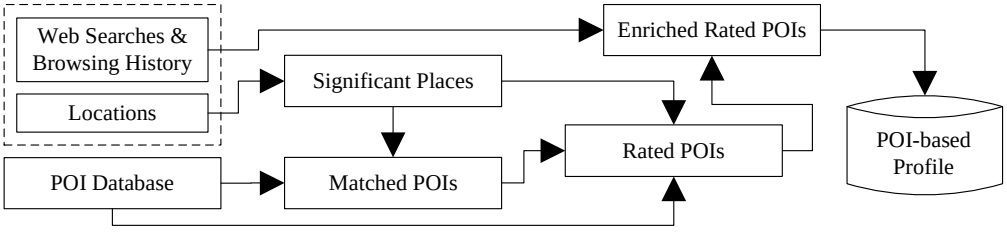

Figure 3. The flowchart of the profiling process.

4.3.1. Significant Places Extraction

The significant places for a user are extracted from his or her raw location traces with the *time-based cluster algorithm* of [34]. The algorithm processes one by one, in one pass, the stored locations and clusters them along the time axis. Each new location measurement is compared to previous locations. Depending on how far the new location is moving away from previous locations, it is considered either part of the previous cluster or the start of a new cluster.

The time-based clustering approach is illustrated in Figure 4. Suppose that the user moves from place A to place B. While the user is at place A, the location measurements are within a certain distance (a parameter of the algorithm) of each other and considered to belong to one cluster; in this example, cluster a. As the user moves towards place B, the location measurements move away from cluster a. On the way to place B, a few small intermediate clusters are generated (i_1, i_2 and i_3). Finally, when the user gets to place B and stays there for a while, a new cluster (cluster b) is formed. If a cluster's time duration is longer than a threshold (specified as the second parameter of the algorithm), the cluster is considered to be a significant place. In the figure, clusters a and b meet the distance and duration criteria and are classified as significant places, whereas the other clusters in between are ignored.

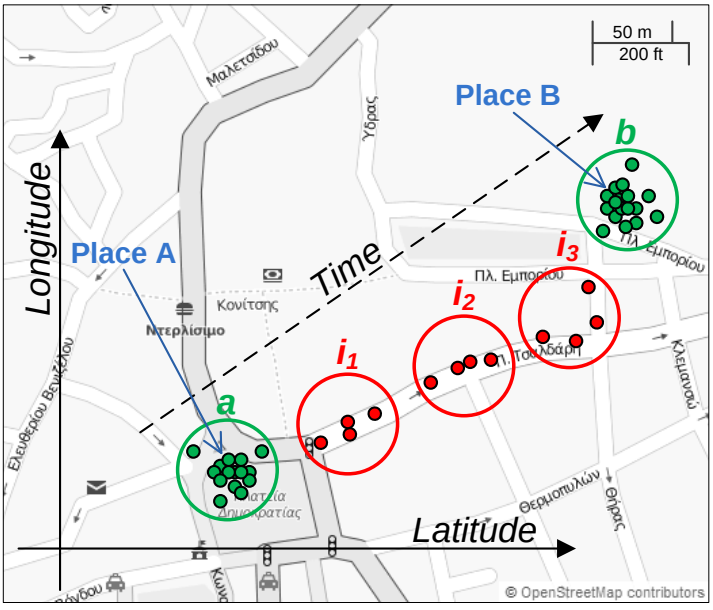

Figure 4. An illustration of the time-based clustering algorithm.

The default parameter values used in the prototype implementation of Pythia are 30 m for the distance threshold and 30 min for the minimum time duration threshold. For each significant place, the extraction process also estimates the number of visits, the duration of each visit and the arrival date and time of each visit.

4.3.2. POI Matching

After identifying significant places from the location traces of a user, these significant places are matched with POIs of the POI database. More precisely, given a significant place c, for each POI p_i within a distance of at most $d_{\max} = \max(\text{average accuracy}, 20\,\text{m})$ from c, we calculate a score:

$$M(p_i) = a \cdot \frac{1}{d(c, p_i)} + (1 - a) \cdot \pi(p_i) \quad (1)$$

where "average accuracy" is the average estimated accuracy of the locations in the cluster of a significant place, $d(c, p_i)$ is the distance between c and p_i and $\pi(p_i)$ is a popularity measure of p_i (e.g., the rating of the POI in Foursquare). A usable and effective value of the weighting coefficient a has to be experimentally estimated; its value does not only depend on the relative importance of distance and popularity, but also on the units used for measuring distance (e.g., meters, kilometers, miles, *etc.*) and on the range of the popularity values $\pi(p_i)$. The significant place is matched to the POI that achieves the highest score.

Note that even though each of the initial significant places is in some sense important to the user, not necessary all of them are POIs of general interest. Thus, some of them will not be matched with POIs in the POI database. Only significant places that are actually matched to POIs play a role in the POI-based profile of the user.

We underline that the POI matching procedure is executed at the user-side, by the application client running on the mobile device or the browser of the user. The client application retrieves the POIs in the area of interest from the central POI database of the Pythia system. To avoid significant leaks about the locations the user is interested in, the client retrieves the related POIs in chunks that correspond either to predefined regions of counties or to predefined divisions of the geographical area (e.g., 30×30 km squares). This way, only a wide area containing the actual location of interest of the user is disclosed to the POI collection framework of Pythia.

4.3.3. POI Rating

Finally, a rating $R(p_i)$ for each matched POI is calculated in an automated way. This rating is an estimation of the user's interest for the POI. We use the five-point Likert scale rating of the Contextual Suggestion Track of TREC 2013, that is each rating is an integer in $\{0, 1, 2, 3, 4\}$, where 0 means strongly uninterested, 2 means neutral and 4 strongly interested. Given a user, first, we calculate for each matched POI in his or her profile an index:

$$f(p_i) = \log(m(p_i))/\mu(p_i) \quad (2)$$

where $m(p_i)$ is the number of the user's visits to p_i and $\mu(p_i)$ is the average number of visits per user to the specific POI, taking into account only those users who have visited the POI p_i at least once. The average number can be calculated from public data from the place search engine.

The value of $f(p_i)$ represents how much more (or less) frequently the user visits p_i, with respect to the rest of the users, normalized in the scale of 0–4. Using $m(p_i)$ to directly estimate a user's preference on a POI p_i would be inadequate in several cases. For example,

1. the user could be very outgoing, or
2. the user could be running Pythia's content collector for a longer period and inevitably have higher $m(p_i)$ values on all POIs, or
3. p_i could be a popular local attraction with a high visitation rate; thus, a high value of $m(p_i)$ is not necessarily indicative of the user's preference.

Therefore, in order to create a metric that properly reflects the user's preference over p_i, it is appropriate to perform a normalization of $m(p_i)$ on two scales: one over the POI's visitation rate of

other users, which confronts a situation like 3, and one over the user's own check-in rates for all of the places her or she has been, which handles Scenarios 1 and 2.

The first kind of normalization is achieved by dividing with $\mu(p_i)$ and is depicted in Equation (2), whereas the second one is described next. Let p_H and p_L be the POIs with the highest and lowest index, respectively. Then, p_H is given a rating of four, p_L a rating of zero and each of the other POIs a rating proportional to its index value. The outcome of the POI matching and automatic rating algorithms described is the POI-based profile of the user, *i.e.*, a set of rated POIs.

Web searches and browsing history: The basic POI-based profile obtained from the location traces of the user can potentially be enhanced by exploiting the browsing history and the web search logs of the user. More precisely, we propose to use these personal data to adjust the POI ratings of the basic profile. For the adjustment, we apply an information retrieval approach. The terms of the web searches and the titles of the browsing history are used to generate a query, which is then submitted to the collection of descriptions of the matched POIs. Depending on how high each POI is ranked, the score of the POI can be adjusted by a small factor. Understandably, this procedure may introduce additional noise into the POI ratings, and for this reason, it is used only for small adjustments. Our current approach is based on rough heuristics, which have not been properly evaluated yet; thus, we do not present them here. We plan to further elaborate on this matter in our future work.

4.4. Contextual Suggestion Algorithm

Given the POI-based profile of a user and a location of interest, the Pythia system returns a set of suggested POIs that are expected to be of interest to the user. The underlying algorithm that makes the POI selection is based on a Rocchio-like relevance feedback method [35,36], and its main steps are depicted in Figure 5. First, the user's POI-based profile is used to generate/train a text query. Then, the generated query is used to score and rank all candidate POIs using their textual representations.

The model is implemented in three main steps:

Step 1: Indexing all candidate POIs. In the first step, a text document is built for each candidate POI. The document is a concatenation of the title, description and categories of the POI; this information is likely to implicitly identify the type of activity that the user was performing at this POI. The description consists of a collection of terms that describes the POI, and it is taken by the POI's web page. The POI documents are indexed with with Indri (the search engine of the Lemur Project [37]) v5.5, using the default settings and Krovetz stemming [38].

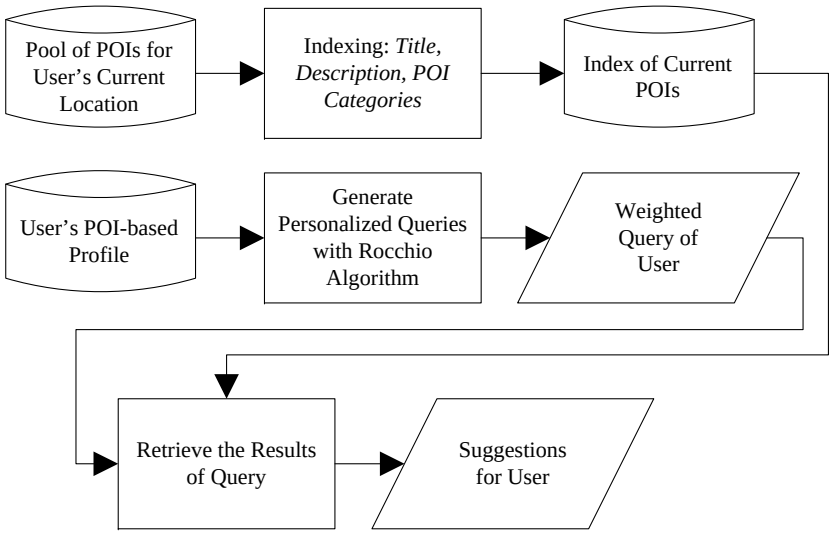

Figure 5. The Rocchio-like contextual suggestion model.

Step 2: Building a query in a Rocchio-like relevance feedback fashion. A personalized weighted query is generated from the POI-based profile of the user using a Rocchio-like relevance feedback method. This query represents the user's preferences. First, a text document is built this time for each POI in the POI-based profile of the user. Let N be the number of distinct terms appearing in these documents. Each POI p_i of the POI-based profile is rated and can be used as a training POI. Let $P_i =< w_{i,1}, \cdots, w_{i,N} >$ be a weighted vector representing p_i, where $w_{i,j}$ is the weight of term j in P_i; we use a standard tf-only (*i.e.*, term frequency) logarithmic weighting: $w_{i,j} = log(1 + f_{i,j})$, where the $f_{i,j}$ is the frequency of term j in the textual representation of POI i. Then, the trained weighted query of the user is the vector:

$$Q = \sum_{j=0}^{4} \left((j-2) \frac{1}{|S_j|} \sum_{i:p_i \in S_j} P_i \right) \tag{3}$$

where the S_j is the set of all POIs in the POI-based profile of the user having a rating of $j \in [0,4]$. In other words, we take the centroids per rating j, multiply them with the normalized rating $j-2$ (so that the neutral rating's centroid, *i.e.*, $j = 2$, does not contribute anything; it is zeroed) and then add the centroids in a Rocchio relevance feedback fashion. All of the terms of the weighted query Q that have a weight less than or equal to zero are excluded from the query. The weight of every term is included in the query by using the Indri Query Language and has, e.g., the following form:

```
#weight( 3.0 museum 2.7 art ··· 0.1 nice)
```

Step 3: Retrieving suggestions. The personalized weighted query Q of the user is submitted to the index generated in Step 1 for the POIs near the user's current location. The search engine is again Indri v5.5 with the default (LM) retrieval model. The results of the query, with a possible cutoff threshold (e.g., top 50 results), are the suggestions to the user for the specific location.

5. Prototype Implementation

To evaluate the feasibility, usability, efficiency and effectiveness of our approach, we developed a prototype of Pythia that implements the proposed privacy-preserving user-centric architecture. In the current status of our implementation, the POI collector and the content collector are fully implemented, and the POI-based profiler and the contextual suggestion engine prototypes are only implemented as web applications. The components of the current prototype are implemented as a collection of standalone applications, which communicate through the personal cloud storage of the user. At this stage of development, we consider the maturity level of our implementation with respect to the Technology Readiness Level (TRL) [39] scale to be somewhere between TRL5 and TRL6.

5.1. Content Collector Implementation

The content collector is a mobile application requiring Android 4.0 or later. It consists of a main *Activity* that presents the collected data to the user, a background *Service* that is coordinating other services to collect data, an *SQLite* database that locally stores the collected data and a series of *AsyncTasks* that perform diverse jobs for the *UI* of the application (Figure 6).

The main features of the current implementation are:

- The *Service* collects the personal data: *web searches*, *browsing history* and *location traces*.
  - *Web searches:* A *ContentObserver* for the URI (Uniform Resource Identifier) *content://browser/searches* is used.
  - *Browsing history:* A *ContentObserver* for the URI (Uniform Resource Identifier). *content://browser/bookmarks* is used.

  - *Location traces:* The *AlarmManager* is used to periodically wake up a new service, which uses a *LocationListener* to retrieve the current location from the GPS provider, the network provider or both.

- The application can export the data in JSON format and in an encrypted form (AES) to the file system (SD card) and the personal cloud storage (Google Drive).
- The application can export the location traces to the SD card in GPXor GeoJSON format; it can backup and restore the *SQLite* database to/from the SD card, *etc*.

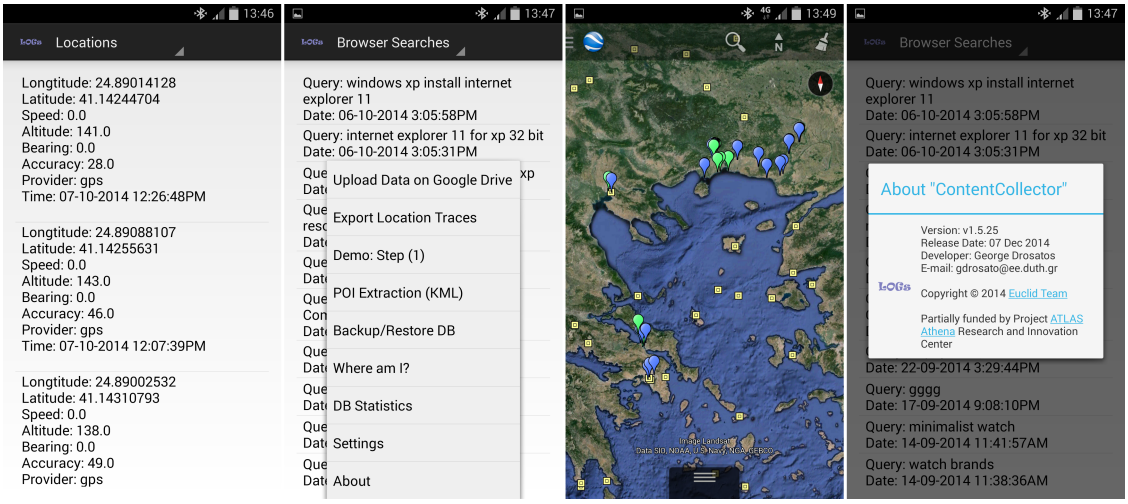

Figure 6. Android app screenshots.

5.2. Other Components

The POI database prototype is built on a PostgreSQL database server [40] and contains the POIs of a number of representative regions from several countries. The PostgreSQL Database Management System (DBMS) was selected because it can natively support geographic data and geospatial operators, features that fit the nature of our dataset. The schema of the POI database has a group of tables for the POIs collected from Foursquare and Google Places, a group of tables for the aggregated POIs and a hierarchy of POI categories (663 categories with 10 parental categories). The service that updates the database is implemented with PHP scripts, which are scheduled by Linux Cron (a time-based job scheduler).

The server-side web API of the POI database is also implemented in PHP and provides the results in JSON format. It supports a broad set of publicly-exposed endpoints to access the database contents, from simple functions to retrieve data about a specific POI to more complex queries, like the one below.

- Searching for POIs in a specific (indexed) area by optionally specifying a multi-term, weighted query for use in the ranking. Sorting is performed using a convex combination of the POI score and the BM25 [41] metric (if a query was provided). An example query is the following:

```
/api/search.php
 ? region = Cephalonia
 & q    = { "beach": { "weight": 2 },
          "bar" : { "weight": 1 } }
 & limit = 10
```

The POI-based profiler is implemented as a JavaScript application that is launched by a web page and is executed at the user-side. The profiler retrieves the raw personal data of the user in encrypted form from the personal cloud storage, using the Google Drive credentials, and decrypts the data with the decryption password of the content collector. The profiler generates the POI-based profile with

the rated POIs and stores it locally. The user can optionally view the profile and make adjustments to the list of POIs and their ratings.

The contextual suggestion engine is also implemented as a JavaScript application that accepts as input the local profile of the user and the current area of interest. The output is a GeoJSON file with the personalized suggestion of the top N POIs for the particular area. This GeoJSON file is used to visualize the results within a map in the web UI (Figure 7).

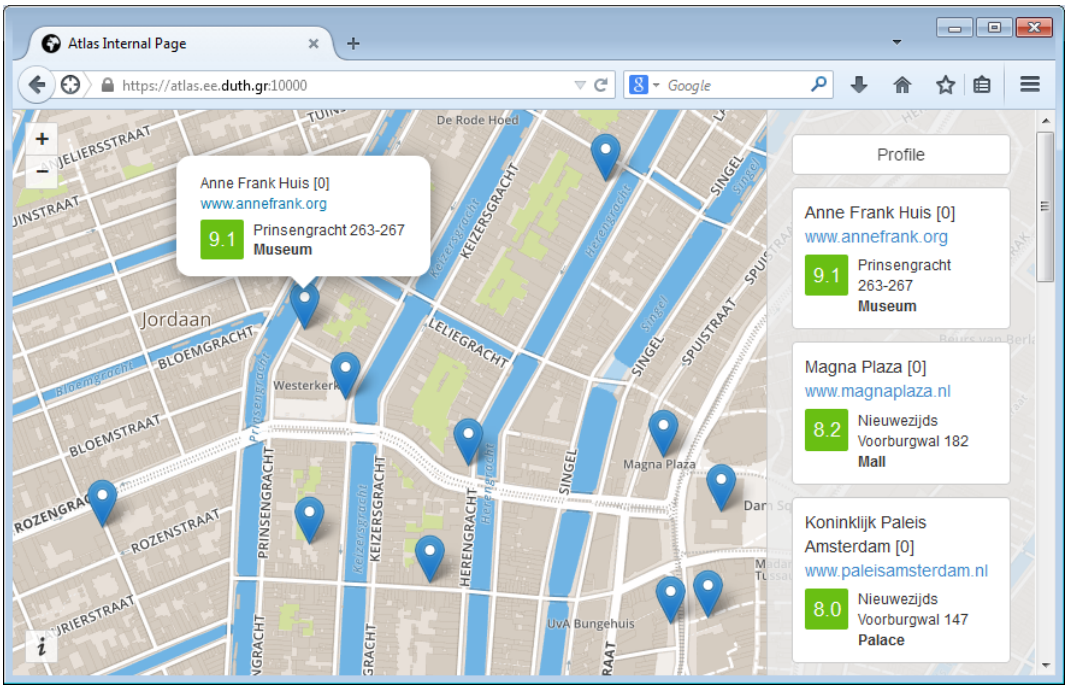

Figure 7. Contextual suggestion web UI.

5.3. Compromises in the Current Prototype

There are compromises in the current Pythia prototype with respect to the planned functionality. Parameter α of Equation (1) is currently set to one because initially, the content collector had no access to data about the popularity of a POI. This issue has now been resolved, but the experiments were already executed for $\alpha = 1$. Due to technical difficulties, the web version of the Rocchio-based algorithm is not absolutely identical to the original algorithm of TREC 2013, and the browser history and web search data are currently not used in the algorithms of the contextual suggestion engine. In both cases, the full implementation was not deemed mature enough.

6. Evaluation and Feedback

The evaluation of Pythia is a challenging task. The system features several functionalities, has to be installed on devices of volunteers and be active a sufficient period of time in order to accumulate at least a minimum amount of data and then manages and processes the personal data at the user-side in a privacy-preserving way. Consequently, it seems very difficult to obtain a large dataset with raw input from all components of the system. Instead, we applied a component-wise evaluation of the critical components of the system and, in addition, asked volunteers for their impression about the whole application. More precisely, we evaluated or obtained feedback on four distinct aspects of the current prototype of the system: the stability and resource consumption of the mobile app, the contextual suggestion algorithms, the POI matching algorithms and the impressions of an admittedly small set of volunteers who agreed to install the application on their Android device and let it record (locally) their activities for some time period.

6.1. Mobile App

The main application component running on the mobile device is the content collector service, which is implemented as a background task. Based on our experiments and the feedback from the volunteers that installed the content collector (see Section 6.4), the app did not cause any noticeable performance degradation on the smart devices. Moreover, the app did not cause any significant increase in energy consumption for the default location sample rate of one location every five minutes.

In particular, the battery usage data of the devices showed for smartphones with low overall usage an energy consumption of about 5%, whereas for smartphones with heavy usage or power-hungry apps, like the Facebook client, the relevant energy consumption of the content collector app was lower; in many cases, the relevant consumption was so low that the app was not even listed in the battery usage data. In any case, none of the volunteers using the content collector reported any case where the energy consumption was an issue. Naturally, the energy consumption grows proportionally to the location sampling frequency, and as such, we tweaked the default settings to allow accurate measurements while minimizing the energy cost. The rest of the functionalities of Pythia, at least in its current version, are executed on demand. For example, generating a profile can take from a few seconds to about one minute for data of two years on a mid-range modern smartphone. If we assume that a user queries the recommendation system only once in a while, the corresponding energy consumption is not an issue.

6.2. Contextual Suggestion

The proposed Rocchio-like model was evaluated in the Contextual Suggestion Track of TREC 2013 [13], where we (the DUTH (Democritus University of Thrace) group) ranked with this algorithm among the best groups of the 15 participants. The track's goal was to investigate POI-search techniques considering as context only the user's location, as well as user interests via personal preferences and past history. In other words, the track focused on one situation: a user with a mobile device with limited interaction, but some sort of a user profile; who is in a strange town and is looking for something to do. There is no explicit query; the implicit query is: *Here I am, what should I do?* The above situation matches very well Pythia's goals.

More precisely, DUTH participated in the "open web" category of the track, *i.e.*, set to find interesting POIs from the full web instead of a smaller controlled data collection. This category was more challenging since it involved processing of big heterogeneous data, requiring computational and network-use efficiency. We presented an approach for context processing that comprised a newly-designed and fine-tuned POI data collection technique, a crowdsourcing approach to speed up data collection and two radically different approaches for suggestion processing (k-NN based and a Rocchio-like).

In the context processing, we collected POIs from three popular place search engines, Google Places, Foursquare and Yelp. The collected POIs were enriched by adding snippets from the Google and Bing search engines using crowd-sourcing techniques. In the suggestion processing, we proposed two methods. The first submitted each candidate place as a query to an index of a user's rated examples and scored it based on the top k results. The second method was based on Rocchio's algorithm and used the rated examples per user profile to generate a personal query, which was then submitted to an index of all candidate places.

The official track evaluation showed that both approaches are working well; especially the Rocchio-like approach was the most promising, since it scored almost firmly above the median system and achieved the best system result in almost half of the judged context-profile pairs. Consequently, the Rocchio-like method was employed in Pythia, which, due to the analytical nature of its main formula (in contrast to iterative learning methods), is also light-weight in computational resources. In the final TREC system rankings, DUTH was found to be the second best group in MRR (Mean Reciprocal Rank) and TBG (a modified Time-Biased Gain) evaluation measures and the third best group in Precision@5 (Precision at Rank 5), out of 15 groups in the category in which we

participated. More details on our methods and the description of the track (goals, datasets, evaluation measures, *etc.*) can be found in [12,13], respectively.

6.3. POI Matching

The implicit creation of the POI-based user profiles is based on the automatic identification of the POIs that the users visit and the inference of the user ratings for each of these POIs. The whole approach is built upon a POI matching procedure that processes the location trace of the user and identifies POIs that the user has visited.

In order to get an indication of how effective the current POI matching procedure is, we focused on the lists of POIs that are generated by the POI matching procedure and checked which of these POIs the users actually visited. More precisely, we asked the owners of the two largest profiles of a field experiment (Section 6.4) to assess the correctness of the list of matched POIs in their profiles. The corresponding statistical results are presented in Table 1.

Table 1. POI-matching results for two large user profiles with data for a period of about two years: from 8 January 2014–17 February 2016.

Data and Results	User Profile 1	User Profile 2
GPS Locations	112,557	84,167
Significant places	4305	2451
Unique Significant places	709	401
Unique POI-matches	139	105
Correct POI-matches	99	63
Accuracy	71%	60%

The collected data cover a period of more than two years with two, by now, mid-range smartphones. User 1 used a Samsung Galaxy S4, while User 2 initially a Galaxy S4, which was later upgraded to a Galaxy S5.

We consider the POI matching results very promising. Even though they certainly contain noise, the overall procedure seems to work surprisingly well; the accuracy is fairly high, and definitely higher than we expected for a prototype. Moreover, there are several reasons why these numbers can be improved. Some of the false POI matches can be attributed to inaccurate or incomplete POI data of the POI search engines for certain areas. In certain geographic areas related to the experiment, the POI data of the POI search engines is still not very accurate. One should expect the amount and the quality of the POI data offered by the POI search engines to constantly improve within the near future. Another observation is that some of the noise in the POI matching procedure follows specific patterns. For example, the current POI matching procedure generates many false matches near the places that a user visits on a daily basis. We believe that we can overcome such issues with a combination of appropriate heuristics and fine-tuning of our algorithms.

Finally, we note that we have no formal procedure for assessing the quality of the automatically-inferred POI ratings yet. As a first step in this direction, we present user feedback obtained from a field experiment with a group of volunteers who used the Pythia application.

6.4. User Experience Feedback

An inherent problem in many privacy-enhanced solutions is the privacy *vs.* utility tradeoff, that is balancing the measures taken to protect privacy against the deterioration of the utility and the usability of the system. We have, by design, ensured the privacy-preserving properties of our system and its non-invasive operation. To obtain user experience feedback about the usability of the system and the quality of its contextual suggestions, we designed a field experiment and started it in January 2015. In total, 26 users volunteered to evaluate the current Pythia system, and 17 of them have

installed the content collector (which is the first step to start using Pythia). The study needs to run for at least a few months, for the personal profiles to accumulate a significant amount of personal data. Nevertheless, we asked the users to anonymously answer a questionnaire based on their preliminary impressions about the system. The summary results from a total of 12 questionnaires is shown in Table 2. In this case, too, a five-point Likert scale is adopted, as in TREC 2013: 0 is the lowest rate, 2 is neutral and 4 is the highest rate.

Table 2. A summary of the user feedback questionnaire results. Responses are expressed in a five-point Likert scale, with 0 for disagreement and 4 for agreement. Below is the average of 17 respondents.

Question	Rate
Content collector (Android App)	
Is the application easy to install and use?	3.42
Is the application stable?	3.50
Does it consume energy?	2.83
Profile Generation (Web UI)	
Does the system identify the correct POIs?	2.83
Are the automatic rates correct?	2.67
Contextual Suggestions (Web UI)	
Do you like the recommendations of the automated profile?	2.75
Do you like the recommendations of a manually defined profile?	3.00
Would you like a hybrid mode automatic + manual profile definition?	3.08
General Questions	
Do you believe that privacy protection is important?	3.83
Do you believe that privacy protection in tourism applications is important?	3.42
Do you think the overall Pythia application is useful?	3.33

The main objective of the experiment was to collect user experience feedback using a small number of simple questions about the overall user experience. In particular, the users were asked to respond about the stability of the application (good stability implies higher credibility for the rest of the answers of the users), any noticeable increase in energy consumption (due to the user-centric architecture, certain computations are executed on the user platform), the quality of the profiler and the contextual suggestions (to ensure that the measures taken to enhance privacy did not degrade the utility of the system) and the overall impression about system and the importance of privacy in such applications.

From the results, it is evident that the overall impression of the users is positive, since all questions achieved on average above the neutral value of two. The content collector, which is the most mature of the Pythia components, performs well with respect to usability and efficiency and is very effective with respect to its tasks (collecting data). The web-based automatic profile generation tools achieve lower, but still above neutral scores. The prototype shows that Pythia is feasible, while the other criteria, *i.e.*, usability, efficiency and effectiveness, could be further improved. Similarly, the implementation of the contextual suggestion engine also provides room for improvements. The POI collection framework is not visible to the users and could not be evaluated in the questionnaire. From our experience, however, the POI collection framework, just like the content collector, is a mature component of the Pythia system (it is up and running without problems since the spring of 2014) and does its job well. Finally, the general questions show that the users value privacy in tourism-related applications almost as high as their general privacy protection and that a system like Pythia is important for tourism.

Admittedly, the number of users in our experiment is very small in order to draw conclusive results. Nevertheless, we still consider the feedback that we obtained very valuable, at least in this stage of the development of the system. Of course, running a large-scale user study on Pythia, once

the system has reached a higher technology readiness level, for example TRL7 (as noted earlier, the current implementation is somewhere between TRL5 and TRL6), would be very useful.

7. Discussion and Conclusions

The Pythia system is a privacy-enhanced, non-invasive contextual suggestion system for tourists, with important architectural innovations. The main system components operate in the background without user interaction, and all personal data of the user are kept on his or her own device. The system's architecture, obtained with a privacy by design approach, allows one to use sensitive personal data for personalized suggestions of touristic attractions without violating the individuals' privacy. The resulting recommendations are context-based (*i.e.*, related to the current geographical place of the user), and the whole system operates non-invasively. The protection of user privacy is achieved by placing the profiling and recommendation procedures at the user-side, either on a mobile device or a browser. Finally, we developed a prototype implementation of Pythia and presented preliminary results with user feedback that confirm the feasibility of our approach.

There are two main lessons we learned from designing and deploying Pythia. First is that the system design, especially the architectural choice to keep all user data on his or her own side, ensures a high level of privacy for the users. This is an innovative feature for a mobile contextual suggestion system. The only leakage is the information about which general areas the user is interested. This leakage can be controlled by masking the actual place of interest within a larger area when the application retrieves data from the POI database. This technique, which is known as *generalization* or *coarsening*, achieves (in the context of the Pythia system) a plausible tradeoff between network load and privacy.

The second lesson learned is that not disclosing user data to the service provider affects the nature of the recommendation service. Collaborative filtering approaches, which are very popular in recommender systems, do not seem to fit our settings. However, this drawback of enforcing non-collaborative users can be counterbalanced by the richness of the single user profile.

We faced several technical challenges during the development and testing of the system, and we addressed most of them successfully, including issues related to low energy consumption, application stability and application usability. A major challenge was the implementation of the contextual suggestion engine within a browser, as a web client application. Even though the core contextual suggestion algorithms were already implemented, converting them into web applications proved to have several difficulties. As a result, in the current prototype, these algorithms are slightly modified, and the contextual suggestion results are not of the same quality as the 'mobile' algorithms. This issue will be addressed by further improving the web client implementations of our algorithms.

Overall, the prototype built has demonstrated that the approach taken in the Pythia system is feasible and can lead to effective contextual suggestion systems.

An interesting observation that we made during our interaction with users and potential users of Pythia is that certain users are reluctant to install an application like the content collector on their mobile devices, even if they are assured that their personal data is not sent anywhere. These users were getting anxious because of the simple existence of a profile containing their personal data and, in particular, their web searches and browser history. These indications led us to include in the questionnaire a question to get some feedback about the relative importance of the protection privacy of the three main types of personal data in Pythia. The outcome provided some evidence that users value the privacy of browser history and web searches, at least as high as their location privacy (Table 3). In conclusion, based on the overall feedback from the users that participated in the experiment and answered the questionnaire, but also from the users who did not accept to participate in the experiment, we believe that we have to investigate how to make more users trust such an application.

Table 3. User feedback about the three types of personal data.

Question	Rate
The following data type is sensitive personal data and should NOT be disclosed to service providers	
GPS Locations	3.58
Browser History	3.67
Web Searches	3.67

Another important challenge that we faced is how to evaluate our system. Regarding the contextual suggestion engine, there is strong evidence that the applied algorithms work well because of their good performance at TREC 2013. However, the other half, the automatic profiling algorithms, as well as the effectiveness of the overall system could not be evaluated in the same way. The fact that the algorithms of the systems work on the personal data of users practically excludes the possibility to make a centralized assessment of the effectiveness of the system. One can use questionnaires like we did already or, for example, run A/B testing, where each hypothesis of the system is experimentally tested on random sets of users. An interesting research problem would be to work on a privacy-preserving method for evaluating the effectiveness of such systems.

The development of the Pythia system is continuing. Our current plans are to upgrade the implementation of Pythia to a more complete and stable system, to perform a more extensive evaluation of the complete system and to investigate further applications of Pythia in the context of smart cities and personal e-health systems.

Finally, we are currently examining two distinct directions for enhancing the profiler and the contextual suggestion engine. First, we are looking into how to use additional information from the smartphones to improve the accuracy of the POI matching procedure; for example, in order to distinguish the case where the user stays for a while in front of a POI (e.g., a cafeteria), from the case that the user actually visits the POI. Second, we are investigating how to enhance the user profile with additional information, beyond the matched POIs. For example, we already support the collection of web search and browser history, but do use these data in the suggestion algorithms. If we could manage to use more (sensor) data from the smartphone or even identify user activities at different locations, this could be used to improved both of the above directions.

Acknowledgments: This work has been partially supported by project ATLAS (Advanced Tourism Planning System), GSRT/CO-OPERATION/11SYN-10-1730.

Author Contributions: Conceived and designed the system: Pavlos S. Efraimidis (PSE), George Drosatos (GD), Avi Arampatzis (AA), Giorgos Stamatelatos (GS), Ioannis N. Athanasiadis (INA). Developed and tested the software: GD, GS, PSE, AA. Analyzed the data: PSE, GD, AA, GS. Performed user experiments: GD, GS, PSE, AA. Wrote the paper: PSE, GD, AA, GS, INA.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xiang, Z.; Tussyadiah, I. *Information and Communication Technologies in Tourism 2014*; Springer: New York, NY, USA, 2014.
2. Gavalas, D.; Kenteris, M. A web-based pervasive recommendation system for mobile tourist guides. *Pers. Ubiquitous Comput.* **2011**, *15*, 759–770.
3. Stabb, S.; Werther, H.; Ricci, F.; Zipf, A.; Gretzel, U.; Fesenmaier, D.; Paris, C.; Knoblock, C. Intelligent systems for tourism. *IEEE Intell. Syst.* **2002**, *17*, 53–66.
4. Yang, W.S.; Hwang, S.Y. iTravel: A recommender system in mobile peer-to-peer environment. *J. Syst. Softw.* **2013**, *86*, 12–20.
5. De Spindler, A.; Norrie, M.C.; Grossniklaus, M.; Signer, B. Spatio-Temporal Proximity as a basis for Collaborative Filtering in Mobile Environments. In Proceedings of the CAISE Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS '06), Luxembourg, 5–9 June 2006.

6. European Parliament. Directive 95/46/EC. In *Official Journal L 281*; 1995; pp. 31–50. Available online: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:HTML (accessed on 20 April 2016).
7. General Data Protection Regulation (GPDR). Available online: http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf (accessed on 20 April 2016).
8. Zhan, J.; Hsieh, C.L.; Wang, I.C.; Hsu, T.S.; Liau, C.J.; Wang, D.W. Privacy-Preserving Collaborative Recommender Systems. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2010**, *40*, 472–476.
9. Barkhuus, L.; Dey, A. Location-Based Services for Mobile Telephony: A study of user's privacy concerns. In Proceedings of the 9th IFIP TC13 International Conference on Human-Computer Interaction (Interact '03), Zürich, Switzerland, 1–5 September 2003.
10. Danezis, G.; Domingo-Ferrer, J.; Hansen, M.; Hoepman, J.; Métayer, D.L.; Tirtea, R.; Schiffner, S. Privacy and Data Protection by Design—From policy to engineering. *Cryptogr. Secur.* **2015**, doi: 10.2824/38623.
11. TREC Contextual Suggestion. TREC 2013 Contextual Suggestion Track Guidelines, October 2013. Available online: https://sites.google.com/site/treccontext/trec-2013-guidelines (accessed on 20 December 2015).
12. Dean-Hall, A.; Clarke, C.L.A.; Simone, N.; Kamps, J.; Thomas, P.; Voorhees, E.M. Overview of the TREC 2013 Contextual Suggestion Track. In Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, MD, USA, 19–22 November 2013.
13. Drosatos, G.; Stamatelatos, G.; Arampatzis, A.; Efraimidis, P.S. DUTH at TREC 2013 Contextual Suggestion Track. In Proceedings of the 22nd Text REtrieval Conference (TREC '13), Gaithersburg, MD, USA, 19–22 November 2013.
14. Meehan, K.; Lunney, T.; Curran, K.; McCaughey, A. Context-aware intelligent recommendation system for tourism. In Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), San Diego, CA, USA, 18–22 March 2013; pp. 328–331.
15. Adomavicius, G.; Tuzhilin, A. Chapter Context-Aware Recommender Systems. In *Recommender Systems Handbook*; Springer US: Boston, MA, USA, 2011; pp. 217–253.
16. Ricci, F.; Rokach, L.; Shapira, B. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*; Springer US: Boston, MA, USA, 2011; pp. 1–35.
17. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl. Based Syst.* **2013**, *46*, 109–132.
18. Borràs, J.; Moreno, A.; Valls, A. Intelligent tourism recommender systems: A survey. *Expert Syst. Appl.* **2014**, *41*, 7370–7389.
19. Gavalas, D.; Konstantopoulos, C.; Mastakas, K.; Pantziou, G. Mobile recommender systems in tourism. *J. Netw. Comput. Appl.* **2014**, *39*, 319–333.
20. Refanidis, I.; Emmanouilidis, C.; Sakellariou, I.; Alexiadis, A.; Koutsiamanis, R.A.; Agnantis, K.; Tasidou, A.; Kokkoras, F.; Efraimidis, P.S. myVisitPlanner^{GR}: Personalized Itinerary Planning System for Tourism. In *Artificial Intelligence: Methods and Applications*; Likas, A.; Blekas, K.; Kalles, D., Eds.; Springer: Gewerbestrasse, Switzerland, 2014; Volume 8445, pp. 615–629.
21. Tsekeridou, S.; Tsetsos, V.; Chalamandaris, A.; Chamzas, C.; Filippou, T.; Pantzoglou, C. iGuide: Socially-Enriched Mobile Tourist Guide for Unexplored Sites. In *Artificial Intelligence: Methods and Applications*; Likas, A.; Blekas, K.; Kalles, D., Eds.; Springer: Gewerbestrasse, Switzerland, 2014; Volume 8445, pp. 603–614.
22. Pliakos, K.; Kotropoulos, C. Simultaneous Image Clustering, Classification and Annotation for Tourism Recommendation. In *Artificial Intelligence: Methods and Applications*; Likas, A., Blekas, K., Kalles, D., Eds.; Springer: Gewerbestrasse, Switzerland, 2014; Volume 8445, pp. 630–640.
23. Erkin, Z.; Veugen, T.; Toft, T.; Lagendijk, R. Generating Private Recommendations Efficiently Using Homomorphic Encryption and Data Packing. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 1053–1066.
24. Jeckmans, A.; Peter, A.; Hartel, P. Efficient Privacy-Enhanced Familiarity-Based Recommender System. In *Computer Security—ESORICS 2013*; Crampton, J., Jajodia, S., Mayes, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8134, pp. 400–417.
25. Polat, H.; Du, W. Privacy-preserving collaborative filtering using randomized perturbation techniques. In Proceedings of the IEEE International Conference on Data Mining (ICDM'03), Melbourne, FL, USA, 19–22 November 2003; pp. 625–628.

26. Riboni, D.; Bettini, C. Private context-aware recommendation of points of interest: An initial investigation. In Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Lugano, Switzerland, 19–23 March 2012; pp. 584–589.
27. De Gemmis, M.; Iaquinta, L.; Lops, P.; Musto, C.; Narducci, F.; Semeraro, G. Learning Preference Models in Recommender Systems. *Preference Learning*; Springer: Berlin, Heidelberg, 2011; pp. 387–407.
28. Lieberman, H. Letizia: An Agent That Assists Web Browsing. In Proceedings of the 14th International Joint Conference on Artificial Intelligence—Volume 1, Montreal, QC, Canada, August 1995; pp. 924–929.
29. Chan, P. Constructing Web User Profiles: A Non-invasive Learning Approach. In *Web Usage Analysis and User Profiling*; Masand, B., Spiliopoulou, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1836, pp. 39–55.
30. OwnCloud. Available online: https://owncloud.org/ (accessed on 20 April 2016).
31. Advanced Encryption Standard (AES). Available online: http://en.wikipedia.org/wiki/Advanced_Encryption_Standard (accessed on 20 April 2016).
32. Google Places. Available online: https://developers.google.com/places/ (accessed on 20 April 2016).
33. Foursquare. Available online: https://foursquare.com (accessed on 20 April 2016).
34. Kang, J.H.; Welbourne, W.; Stewart, B.; Borriello, G. Extracting Places from Traces of Locations. In Proceedings of the 2nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, WMASH '04, Philadelphia, PA, USA, 26 September–1 October 2004; pp. 110–118.
35. Rocchio, J.J. Relevance Feedback in Information Retrieval. In *The Smart Retrieval System-Experiments in Automatic Document Processing*; Salton, G., Ed.; Prentice-Hall: Englewood Cliffs, NJ, USA, 1971; pp. 313–323.
36. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: New York, NY, USA, 2008.
37. Lemur Project. Available online: http://www.lemurproject.org (accessed on 20 April 2016).
38. Krovetz, R. Viewing morphology as an inference process. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93), Pittsburgh, PA, USA, 27 June–1 July 1993; pp. 191–202.
39. EARTO. The TRL Scale as a Research & Innovation Policy Tool. In *EARTO Recommendations*; 2014; pp. 1–17. Available online:http://www.earto.eu/fileadmin/content/03_Publications/The_TRL_Scale_as_a_R_I_Policy_Tool_-_EARTO_Recommendations_-_Final.pdf (accessed on 20 April 2016).
40. PostgreSQL. Available online: http://www.postgresql.org (accessed on 20 April 2016).
41. Robertson, S.E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M.; Gatford, M. Okapi at TREC-3. In Proceedings of the Third Text Retrieval Conference, TREC '94, Gaithersburg, MD, USA, 2–4 November 1994.